

MYSQL TRANSACTIONS

<http://www.tutorialspoint.com/mysql/mysql-transactions.htm>

Copyright © tutorialspoint.com

A transaction is a sequential group of database manipulation operations, which is performed as if it were one single work unit. In other words, a transaction will never be complete unless each individual operation within the group is successful. If any operation within the transaction fails, the entire transaction will fail.

Practically you will club many SQL queries into a group and you will execute all of them together as a part of a transaction.

Properties of Transactions:

Transactions have the following four standard properties, usually referred to by the acronym ACID:

- **Atomicity:** ensures that all operations within the work unit are completed successfully; otherwise, the transaction is aborted at the point of failure, and previous operations are rolled back to their former state.
- **Consistency:** ensures that the database properly changes states upon a successfully committed transaction.
- **Isolation:** enables transactions to operate independently of and transparent to each other.
- **Durability:** ensures that the result or effect of a committed transaction persists in case of a system failure.

In MySQL, transactions begin with the statement **BEGIN WORK** and end with either a **COMMIT** or a **ROLLBACK** statement. The SQL commands between the beginning and ending statements form the bulk of the transaction.

COMMIT and ROLLBACK:

These two keywords **Commit** and **Rollback** are mainly used for MySQL Transactions.

- When a successful transaction is completed, the **COMMIT** command should be issued so that the changes to all involved tables will take effect.
- If a failure occurs, a **ROLLBACK** command should be issued to return every table referenced in the transaction to its previous state.

You can control the behavior of a transaction by setting session variable called **AUTOCOMMIT**. If **AUTOCOMMIT** is set to 1 (the default), then each SQL statement (within a transaction or not) is considered a complete transaction, and committed by default when it finishes. When **AUTOCOMMIT** is set to 0, by issuing the **SET AUTOCOMMIT=0** command, the subsequent series of statements acts like a transaction, and no activities are committed until an explicit **COMMIT** statement is issued.

You can execute these SQL commands in PHP by using **mysql_query()** function.

Generic Example on Transaction

This sequence of events is independent of the programming language used; the logical path can be created in whichever language you use to create your application.

You can execute these SQL commands in PHP by using **mysql_query()** function.

- Begin transaction by issuing SQL command **BEGIN WORK**
- Issue one or more SQL commands like **SELECT**, **INSERT**, **UPDATE** or **DELETE**

- Check if there is no error and everything is according to your requirement.
- If there is any error then issue ROLLBACK command otherwise issue a COMMIT command.

Transaction-Safe Table Types in MySQL:

You can not use transactions directly, you can but they would not be save and guaranteed. If you plan to use transactions in your MySQL programming then you need to create your tables in a special way. There are many type of tables which support transactions but most popular one is **InnoDB**.

Support for InnoDB tables requires a specific compilation parameter when compiling MySQL from source. If your MySQL version does not have InnoDB support, ask your Internet Service Provider to build a version of MySQL with support for InnoDB table types, or download and install the MySQL-Max binary distribution for Windows or Linux/UNIX and work with the table type in a development environment.

If your MySQL installation supports InnoDB tables, simply add a **TYPE=InnoDB** definition to the table creation statement. For example, the following code creates an InnoDB table called tcount_tbl:

```
root@host# mysql -u root -p password;
Enter password:*****
mysql> use TUTORIALS;
Database changed
mysql> create table tcount_tbl
-> (
-> tutorial_author varchar(40) NOT NULL,
-> tutorial_count INT
-> ) TYPE=InnoDB;
Query OK, 0 rows affected (0.05 sec)
```

Check following link to know more about : [InnoDB](#)

You can use other table type like **GEMINI** or **BDB** but it depends on your installation if it supports these two types.